Virtual Reality Guide to Hospital Autopsy: Block Dissection of the Renal System

by
Daniel W. Hermansen

A thesis submitted to Johns Hopkins University
in conformity with the requirements for
the degree of Master of Arts

Baltimore, Maryland
March, 2017

Abstract

Hospital autopsies are a highly valuable source of pathological evidence for understanding the progression and nature of diseases. Autopsies can also serve as a healthcare audit, assuring that premortem diagnoses are complete and accurate. However, requests for hospital autopsies have continually declined since 1970. This has resulted in a lack of understanding of the benefits of autopsies among patients and clinicians, and a dearth of professionals trained to competently perform the procedure. With limited access to procedures and professionals qualified to train others, certification has become increasingly difficult. In addition, the training materials currently available are limited and do not allow one to physically practice the techniques involved in the autopsy procedure.

This study documents the methods used in developing the **Virtual Reality Guide to Hospital Autopsy: Block Dissection of the Renal System**. This virtual simulation serves as a training tool to teach proper post-mortem examination technique through immersive user interaction. The simulation covered in this study will serve as the foundation and proof-of-concept for a future expanded program covering each of the five anatomical block dissections involved in a standard autopsy. This study will address the creation of one of those blocks, the renal system, in particular the creation and utilization of 1) a virtual user interface, 2) interactive virtual instruments, 3) accurate virtual models of renal system anatomy, 4) realistic tissue reaction of organic models with instrument interaction, and 5) a demonstration of certain procedural steps performed on the renal anatomy. The simulation was designed for use on the Oculus Rift headset and Touch motion controllers.

Simulation-based medical education is a platform proven to provide immersive, competency based training that improves clinician procedural performance in cases with limited patient access or uncommon procedures. Virtual reality technology allows for repeatable simulation scenarios, which can provide feedback over time to identify trends in user performance. The Virtual Reality Guide to Hospital Autopsy assists in learning complex procedural techniques by simulating realistic representations of tissue reaction and accurate user interaction controls.

Masters Thesis Preceptors for Daniel Hermansen:

Jody E. Hooper, MD, Director of Autopsy Service

David A. Rini, MFA, Associate Professor (Faculty Advisor)

This project reflects a major step in my education and was made possible by several people who donated their time and expertise. To all those involved, I would like to express my immense gratitude for all the help given throughout the process.

David A. Rini, Associate Professor in the Department of Art as Applied to Medicine, and Dr. Jody E. Hooper, Director of Autopsy Service and Assistant Professor of Pathology at Johns Hopkins University School of Medicine, both played active roles in the development of this project as faculty advisor and preceptor, respectively. Their advice and ready availability throughout the course of the project were invaluable to its success.

A sincere thank you as well to the pathology residents who walked me through the series of autopsies that served as the foundational reference for this project.

Many thanks to the following people for their expertise and contributions: Nate Crookston, who took the time from his family and his own PhD studies to develop and teach me the scripts used in the simulation; Emily Shaw, for setting me on the path to the Society for Simulation in Healthcare and the IMSH conference. The inspiration gained there has helped shape this project and will continue to influence all my future endeavors; my MedArt family, Katie Bergdale, Sarah Chen, Nicholas Reback, Li Yao, Eleanor Bailey, and Julia Lerner, for standing together through all the frustrations, critiques, and triumphs in this entire experience; the faculty and staff of the Department of Art as Applied to Medicine, for their patience, kindness, and the incredible depths of talent and experience they provided these past two years; the Vesalius Trust, for their financial support of this project and the many other projects that continue to help and redefine the field of biomedical communications; and finally, and my most heartfelt gratitude to Kim, my beautiful wife, best friend, and soon-to-be mother, and to all the friends and family who have supported me through everything and, without whom none of this would be possible.

Table of Contents

Hospital autopsies are performed to ensure quality control in medicine, to help confirm the presence of disease, for research, and to further educate physicians. Autopsies also increase understanding of the progression and nature of disease; between 1950 and 1983 alone, autopsies helped discover or clarify 87 diseases or groups of diseases (Hill and Anderson, 1996). Even with the advances in imaging technology and techniques, eight to ten percent of autopsies still discover new diagnoses, which represent major discrepancies from premortem findings (Shojania and Burton, 2008).

In recent years, there has been a significant decline in hospital autopsy requests. Prior to 1970, autopsies were performed in forty to sixty percent of cases involving hospital deaths in the United States; that number has now decreased to five percent (Shojania and Burton, 2008). There are several causes for this decline including increased costs and institutional budgetary restrictions, but it is proposed that the greatest contributing factor is a lack of patient and clinician education about the benefits of hospital autopsy (Davies, et.al., 2004). Many hospitals no longer perform autopsies on-site and choose to contract with outside sources to complete the procedure in only the most unique cases. Training hospitals or those attached to medical universities are the primary sources of continued hospital autopsy research and training; however, even some of these only receive one or two autopsy requests per year. The American Board of Pathology currently requires that a resident perform a minimum of fifty autopsies to become certified in combined anatomic and clinical pathology (The American Board of Pathology, 2015). For many pathologists, this may mean traveling to distant institutions that regularly perform hospital autopsies. Much of the initial training occurs through direct observation of senior or attending pathologists that can result in inconsistencies due to variations in procedures and individual pathologist's techniques. All of these factors have led to an insufficiency in the expertise needed to competently perform a hospital autopsy.

Current training materials for hospital autopsy include manuals with illustrations and video tutorials that demonstrate proper technique, but no solution currently exists that allows the student to physically practice the procedure without access to an autopsy case. The techniques used in a hospital autopsy require a level of dexterity, which can be reproduced using current virtual reality simulation technology. Simulation-based learning has been shown to increase procedural and technique

retention through the option of repeatable scenarios that mimic real-life situations (Lateef, 2009). This allows the user to practice and review specific aspects of a procedure without risking damage to a patient, organs, or tissues. Simulation-based learning also outlines and demonstrates a consistent accuracy in the procedural steps to avoid any training inconsistencies and subsequent operative mistakes. The user can also learn at his or her own pace by reviewing troublesome subject areas or complicated techniques until they have achieved a mastery of the subject matter.

Virtual reality (VR) training has been an important component of medical simulation centers for several years, but the cost of specialized simulators is often a prohibitive factor and the consoles are configured to accommodate a specific set of healthcare scenarios. With the recent release of several VR consumer headsets, the technology costs have become much less of a barrier. As the technology continues to improve, developers will continue to create and release a wider range of virtual content covering a variety of subjects, leading to wider adoption of VR technology in hospital and medical training centers.

In collaboration with Jody E. Hooper, M.D., Director of Autopsy in the Pathology Department at Johns Hopkins Hospital, a VR simulation was developed to demonstrate the steps and techniques used in the block dissection of the renal system. The program is intended as a proof-of-concept and model for the development of a more comprehensive program that will cover each of the five body block dissections in a hospital autopsy including: 1) heart, 2) lungs, 3) hepatic block, 4) renal block, and 5) pelvic block. Additional details regarding the nature and appearance of certain pathologies will be added along with increased testing and evaluation functionality. The techniques used to develop this test module are outlined in this study. Also highlighted, are the potential technological opportunities for expansion into other platforms and how some principles covered in this study may apply to other healthcare applications.

Hospital Autopsy

To address the need for an accessible training program for hospital autopsy procedure and technique, a meeting was organized with Jody Hooper, M.D. to discuss the creation of a detailed interactive simulation. After discussing the various training formats, it was determined that the creation of a VR simulation would maximize the user's ability to learn the necessary steps and techniques of the procedure by immersing themselves in an interactive training environment. A list of objectives was developed to guide the project and criteria were set to best demonstrate the effectiveness of VR simulation in autopsy training. The objectives were defined as follows:

- A step-by-step simulation of proper autopsy technique on a virtual model of the renal system.
- An innovative use of VR to explore new methods of procedural instruction technique.
- To investigate the depiction of realistic tissue reaction and effective user interface design.

Details of a hospital autopsy procedure were established through the observation of five autopsies performed within the Johns Hopkins Hospital Pathology Department, as well as a literature review and detailed discussions with Dr. Hooper. The procedural steps were documented and critical anatomical and procedural details were photographed for reference while modeling and programming the simulation. All photographs were taken and archived in accordance with HIPAA guidelines.

An extensive literature review of existing autopsy procedure methods and manuals was carried out and the material was carefully assessed to aid in designing the most accurate and effective procedural simulation possible. The research identified specific divisions within the procedure, which were used to create manageable anatomical sections or blocks to simulate individually. These divisions allow the user to focus on specific areas and steps of the procedure and keeps the virtual reality rendering and processing complexity within an acceptable range.

The renal block was selected to model and simulate because it involves a variety of unique steps and user interactions included in the dissection of the kidneys, ureters, and bladder. The steps of the procedure involving were determined as the following:

1. Vascular structures
    i. Use the scissors to open the inferior vena cava, renal veins, and right iliac veins. Record if thrombus is present
    ii. Open iliac arterial branches, aorta, and renal arteries. Record the amount of atherosclerosis in renal arteries or any pathology.
2. Renal structures
    i. Use the scissors to incise and open the bladder along the anterior midline from base to dome.
    ii. Describe any bladder lesions.
    iii. Probe the ureteral orifices and ureters to insure patency.
    iv. Use the scalpel to make a small incision in the renal capsule to expose the cortical kidney surface.
    v. Peel the renal capsule off the kidney.
    vi. Use the scissors to nick and open the ureter toward the proximal end.
    vii. Run the probe up the ureter into the renal pelvis and through the kidney.
    viii. Use the autopsy knife to bivalve the kidney using the probe as a guide.
    ix. Breadloaf the remaining kidney in the coronal plane at approximately 1 cm intervals.
    x. Measure cortical thickness and record.
    xi. Repeat steps 2iv-2x for opposing kidney.
    xii. Describe any kidney lesions.
    xiii. After organ review, detach kidneys, weigh and record the weights.
    xiv. Sample the kidneys and any lesions, if present.

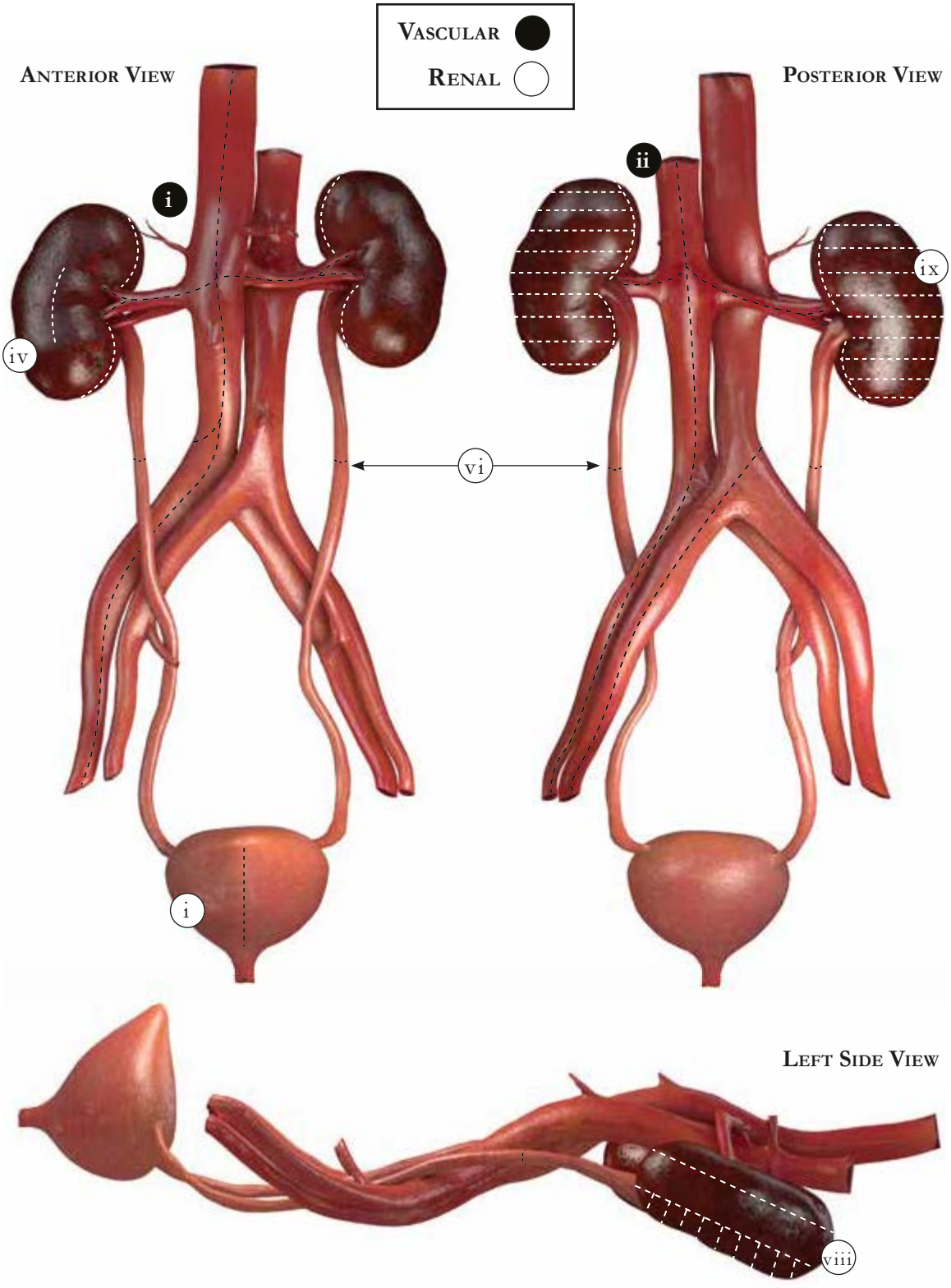

Figure 1. Renal system cutting guide (cutting steps from previous page).

ASSET DEVELOPMENT

All 3D assets were created using Cinema 4D® Student Edition. Instruments, used in the autopsy department at Johns Hopkins Hospital Pathology Department, were referenced for modeling the virtual instruments. Scans of each instrument were taken with a flatbed scanner and used as a guide for the models. The scanned images were attached to perpendicular planes to indicate the instrument dimensions in orthogonal views.

Each instrument was created with a combination of the basic parametric shapes within Cinema 4D®. Once properly aligned with the image guides, the shapes were made editable thus allowing the meshes to be manipulated and joined as necessary. Open and closed states were created for the scissors and forceps.

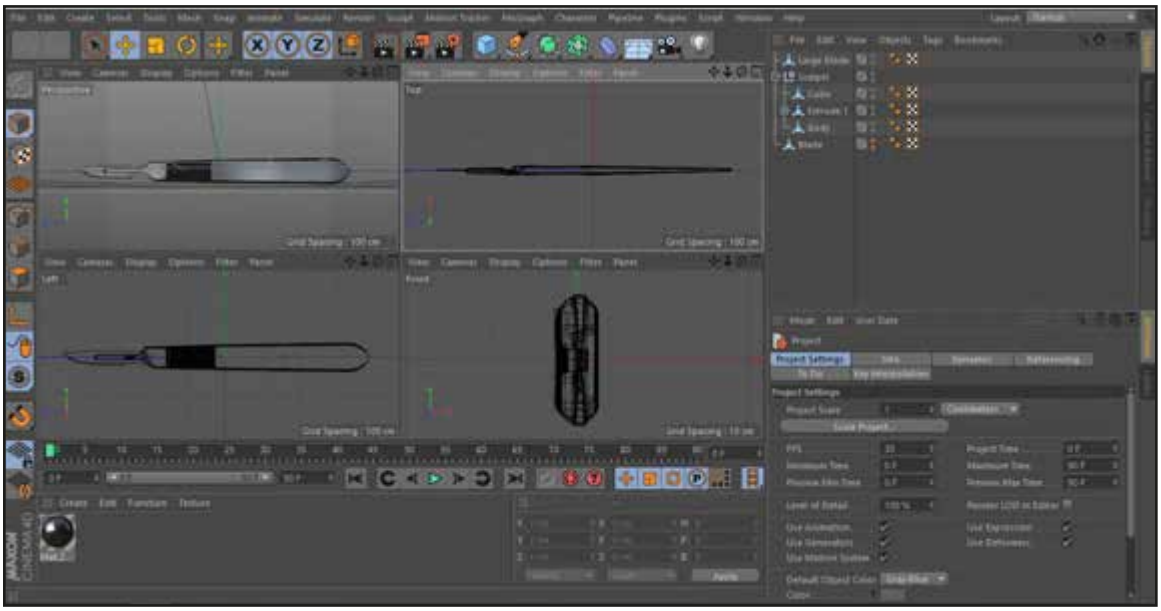

Figure 2. Scalpel model creation in Cinema 4D (text not intended to be read).

The kidney models were created using a "metaball" modifier attached to 3 spheres. Once editable, the sculpting features within Cinema 4D® were used to smooth the surfaces and minor variations were added to the kidney surface to add organic realism. The grooves in the renal pelvis were sculpted with the "pinch" tool taking care not to overlap the mesh.

Once the kidney model was created, a duplicate was made and rotated to the proper orientation, opposite the original kidney. Both models were then duplicated to create the renal capsule surrounding the kidneys. A cloth surface was added to the renal capsule and a positive thickness of 2 cm was added. This created a layer directly against, but separate from the kidney model.

The bladder model was sculpted from a sphere by grabbing the appropriate vertices and pulling them into the desired form. An opening was made in the base of the mesh and the surrounding ring extruded to create the urethra. The model was then smoothed and a "cloth surface" added to give the bladder a wall thickness of 6 cm. A "formula" deformer was added to the model to simulate rugae and the external surface was selected and smoothed.

The aorta, inferior vena cava, and ureters were created using a series of cylinders. The original models had open ends on the vessels and ureters, but this resulted in issues with the soft body simulation within Unity, so the ends were closed with polygons and the appearance of openings was simulated with material mapping. Branching vessels were modeled and connected to the main vessel using the "CV Boole Union/Add" plugin. Any non-manifold geometry or open edges were detected and repaired in modeling mode. Non-manifold geometry results in spastic reactions in simulation; especially when dealing with soft body and cloth simulation and interaction.

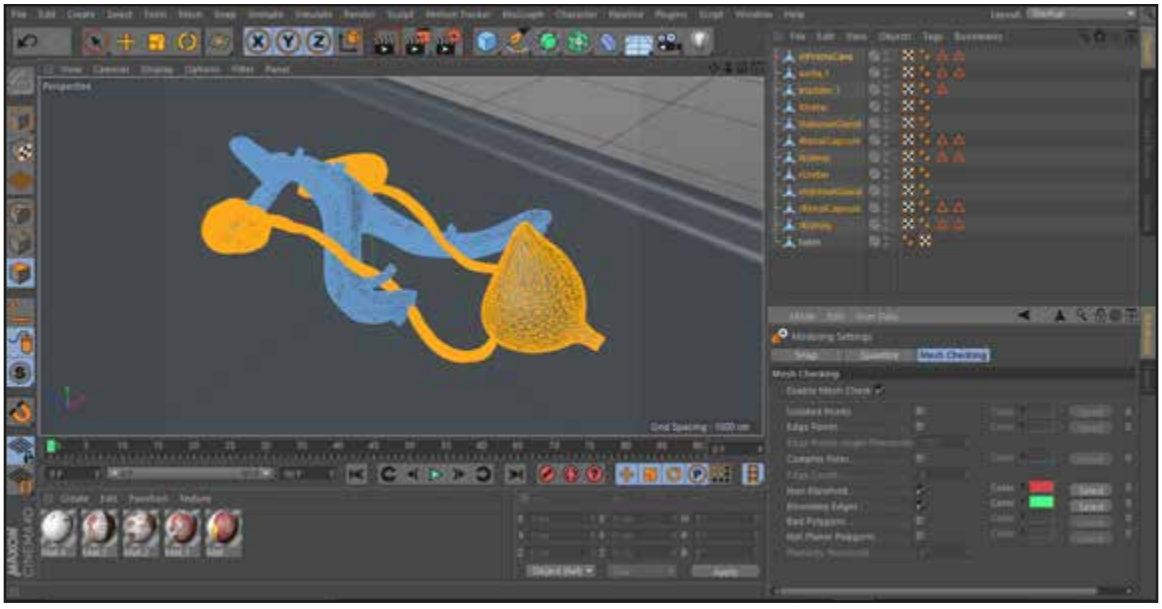

Figure 3. Renal system model and non-manifold geometry correction in Cinema 4D modeling mode (text not intended to be read).

Each aspect of the anatomy was joined into a single model using the "Magic Merge" plugin and any non-manifold geometry was removed. A "polygon reduction" object was added as a child of the model to reduce the geometry to fall within the limits of the uFlex® Unity asset used to simulate the anatomy. A "connect and delete" command was applied to the polygon reduction object and model to create a final model.

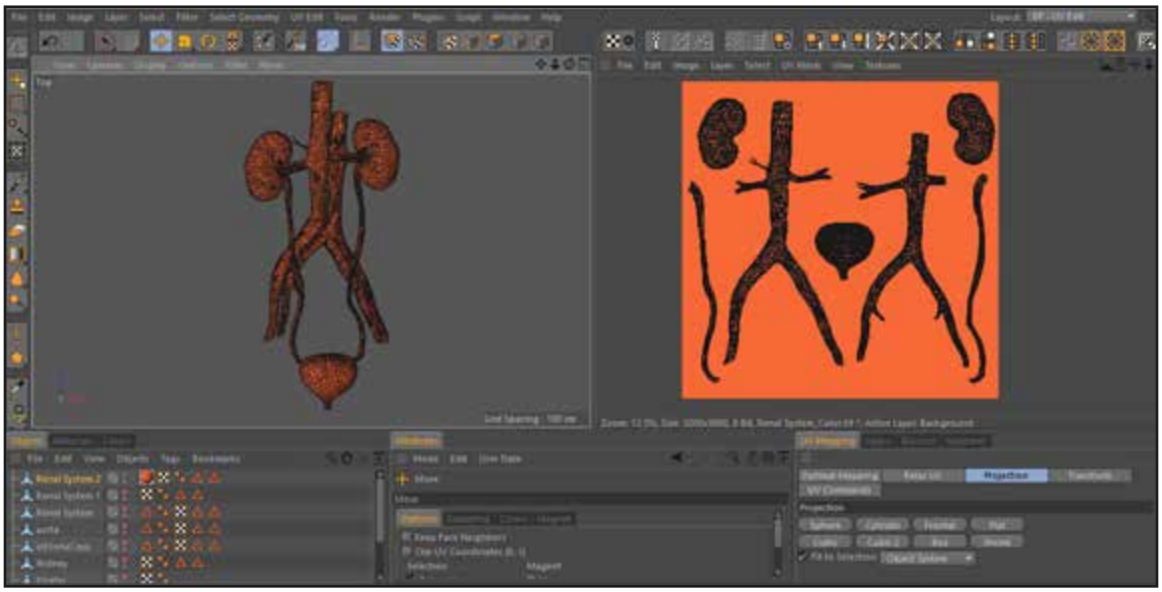

Figure 4. Anatomical elements UV unwrapped in BodyPaint UV Mode (text not intended to be read).

The material and texture for the model were created using the BodyPaint UV mode within Cinema 4D®. With the model selected, the Paint Setup Wizard was used to create color, reflectance, and bump channels with a resolution of 3000. The UV mesh was unwrapped and organized into the individual anatomical structures viewed in the frontal projection format. Each aspect was colored and textured to reflect the surface of their gross anatomical equivalent. The 3D models were exported in Filmbox (.fbx) format to import into Unity game engine.

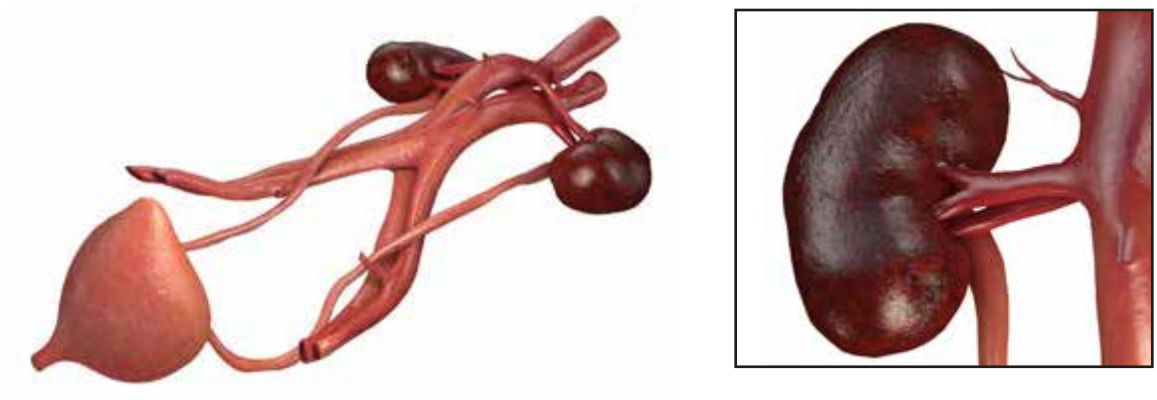

Figure 5. Renal system model with material and texture applied. Kidney texture (right).

Unity Game Engine

Unity® game engine, personal edition, was used to create the virtual reality simulation and integration of the Oculus Rift headset and Touch motion controllers. In conjunction with the release of the Touch motion controllers in December 2016, Oculus® released several software development kits (SDK) to the public to help developers integrate the hardware into simulations. The Oculus Platform SDK, Oculus Avatar SDK, Oculus Audio SDK, Oculus Utilities for Unity 5, and Oculus SDK for Windows were downloaded from the Oculus developer website (Oculus Downloads, 2016).

The Oculus SDKs were imported into the project and, following Oculus' Platform SDK Developer Guide, the program framework was set up (Oculus Platform SDK Developer Guide, 2016). A new 3D project was created and unique application ID generated in the Oculus Developer Center for the application. This ID is retrieved from the API page and is entered under the Oculus Avatar and Oculus Platform menu items along the Unity menu bar to initialize the Platform SDK.

VR support was enabled by opening the Edit menu and selecting Project Settings > Player. In the Unity Inspector window in the Other Settings pane, the Virtual Reality Supported check box was selected and Oculus was added in the Virtual Reality SDKs list.

The Main Camera created at the opening of a new project was replaced with the OVRCameraRig prefab from the OVR asset to link the camera in Play mode to the Oculus Rift headset. From the Oculus Avatar SDK under prefabs, the Local Avatar prefab was selected and placed into the Project window enabling first person use of the Touch motion controllers.

Custom Avatar Hand Poses

Pre-rigged hand models included in the Local Avatar prefab in the OVR Avatar SDK were used for this project. Within the GripPoses sample scene, the hand models were adjusted by rotating each joint to match the hand positions for the proper use of surgical instruments. With the accurate positions selected, each hand model was saved as a new object prefab in the project library.

The "LocalAvatarHandController" script (Appendix C) was created to call the appropriate hand pose into the "Custom Pose" field of the OVR Avatar script on an *AliasGrabOn* event. When the event is called, the script determines if an instrument is being touched using the "VRTK_InteractTouch" script. Each instrument, when picked up, is paired with the appropriate custom hand pose prefab. Right and left "SnapPoints" were created as children of each instrument and added to the "Right and Left Snap Handles" on the "VRTK_Fixed Joint Grab Attach" component of their parent instrument. The "SnapPoint" orientations were configured for the left and right controllers for each instrument allowing the instruments to snap to a specified position and orientation to the active hand.

Figure 6. Custom hand poses with instrument SnapPoints enabled on Local Avatar

User Interface (UI)

The user interface menu was designed using Adobe® Illustrator and recreated in Unity® as a functional interface. Image components of the user interface were exported from Adobe® Illustrator as .png files and added to the Unity asset folder, converted to 2D sprites, and attached to UI image game objects.

The UI menu is contained within a UI canvas. All UI elements are children of the canvas and are controlled by the "EventSystem" component which is automatically generated when the canvas is created. Panels were generated to parent specific portions of the menu and each was equipped with a ToggleGroup component. The toggle buttons within each panel are joined by the ToggleGroup of their parent, allowing only one toggle element to be on at a time. The toggle button appearance is managed by the "ButtonStyler" script (Appendix B) which specifies the normal, highlight, and pressed states of each toggle. The normal toggle state was created using a frame sprite to display only the edges of the toggle. In its highlighted state, the sprite switches to a blank state that fills the toggle with the predetermined color and changes the text to white. When pressed, the toggle adopts the highlighted state permanently until another toggle within the same ToggleGroup is selected.

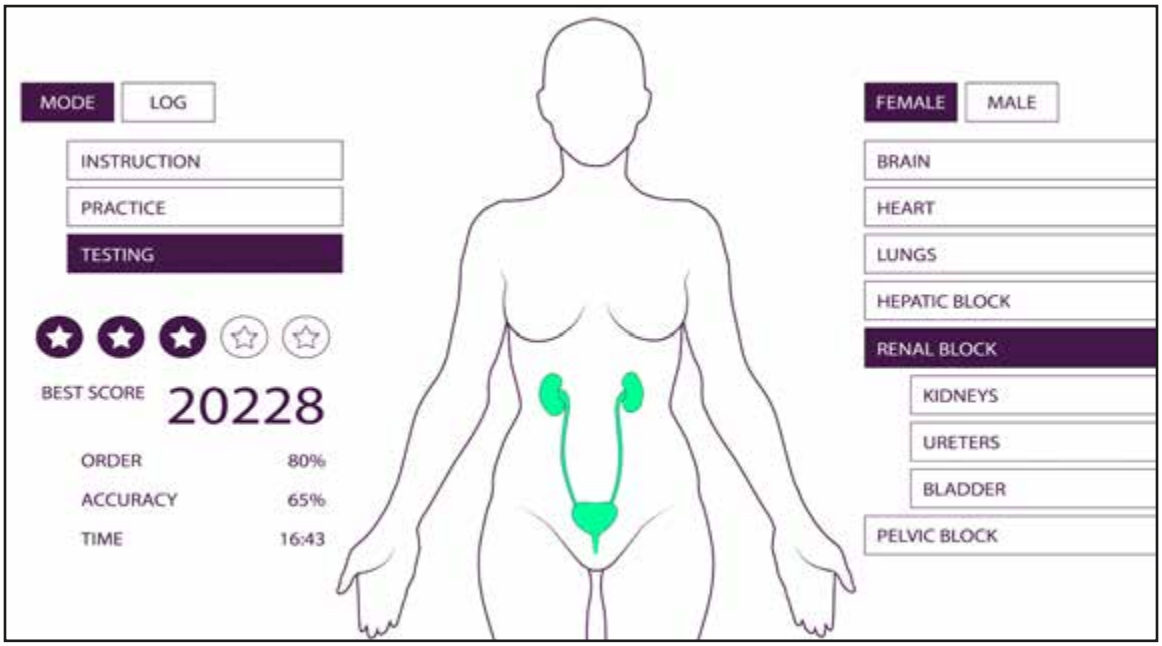

Figure 7. User interface template created in Adobe Illustrator

On start, the "Overlord" script (Appendix A) identifies all components of the user interface and inactivates all panels except the initial toggle button options. When pressed, each toggle button triggers the "Overlord" script to activate one or more of the inactivated elements within the canvas. If a different toggle button within the ToggleGroup is selected, then the UI elements associated with the previous toggle button are inactivated and a new set of UI elements is activated according to the new selection.

With the UI menu elements organized, a "CurvedUI" script component from the Unity asset store was attached to the canvas and its children allowing the canvas to bend. The canvas was curved to bring the far edges toward the user, thus making the UI elements more accessible.

Renal System

The renal system underwent several iterations throughout the course of this project (see Discussion below). The components of the final render of the renal block were joined into a single game object and converted to a softbody object using the uFlex asset downloaded from the Unity asset store. The uFlex system calculates the vertices of a given mesh and creates a series of particles that are organized into shapes that affect a group of vertices, each with their own individual properties. These particles are then joined by a series of bones or hinges to define how they interact with their adjacent particles. The entire system of particles is then assigned a skinned mesh surface to approximate the surface geometry of the system based on the particles position and interconnecting relationships.

As a single softbody object, the renal anatomy deformations were primarily defined by object density. The kidneys are the thickest components, therefore, exhibit the slightest deformation. The thinnest elements, the ureters and bladder walls, conversely demonstrate the greatest deformation with user interaction.

Scripting

All coding was done in the C# language within Unity® game engine and Microsoft Visual Studio®. Assets were downloaded from the Unity asset store to form a scripting base for the project.

Virtual Reality Toolkit® (VRTK) is a collection of virtual reality scripts and concepts that provide a basis for solutions involving interaction with usable objects and user interface elements (VRTK - Virtual Reality Toolkit, 2016). After downloading and importing VRTK from the Unity asset store, the "VRTK_SDKManager" script was added to the scene and attached to an empty GameObject. In each of the public domains under SDK Choices, Oculus VR was selected and the "Auto Populate Linked Objects" button was selected to find and attach all the relevant linked objects.

A "VRTK_InteractableObject" script component was attached to each instrument allowing the user to interact with them via the motion controllers. To increase the user's sense of when an interactable object was being touched and was ready to grab, a blue highlight color was added that overrides the global textures of the instruments. The highlight color remains until the instruments are interacted with or the user is no longer touching them. This required an additional script component, "VRTK_InteractTouch", to be added to the controllers, which provided a collider to recognize the proximity of each controller to an interactable object.

The "VRTK_InteractGrab" script was added to each controller object along with the "VRTK_ControllerEvents" script to listen for controller button events for grabbing and releasing the interactable instruments. It listens for the *AliasGrabOn* and *AliasGrabOff* events to determine when an intractable instrument should be grabbed or released. This also required the "VRTK_InteractableObject" script on each instrument to have the "isGrabbable" flag set to true. Each instrument was also fitted with a collider and rigidbody component allowing them to be picked up and moved in the virtual environment. The break force on the interactable object scripts was set to Infinity to keep the instruments from breaking free from the user's grip while the grip button was being held.

The "PickupsTrigger" script (Appendix D) was written to allow the forceps to attach to and pick up the organs of the renal block by their individual particles. The script calls on the *FlexProcessor* to

recognize when a sphere collider placed at the tip of the forceps is in contact with particles within the anatomy models. When the user presses the index trigger, the open state of the forceps is deactivated and the closed state becomes active effectively pinching the forceps shut. The *trigger pressed* event also checks to see if the sphere collider is in contact with any particles within the renal anatomy; if true, then particles within its radius become locked to the tip of the forceps. As the user moves the forceps in the virtual space, the position of the sphere collider is continuously updated and the attached particles follow in turn. An "aggregate" function was added to apply the position changes to each successive group of neighboring particles to follow the grabbed particles, but react according to the limitations of the soft body hinges between them. This allows the anatomy to twist and deform in a realistic manner around the point of attachment. The particles are also set to recognize each other and result in self collisions preventing the anatomical elements from intersecting with themselves. The attachment point to the forceps remains active until the index trigger is released at which point the joint is destroyed.

A DropZone gameobject was created with a trigger collider situated above the floor of the simulation. The "OnDroppedTool" script (Appendix E) was created and attached to the DropZone gameobject to recover any dropped instruments to their original position on the tray. The original position and rotation of each instrument is captured at the start of the simulation. When an instrument collider body enters the DropZone, a *onTriggerEntered* event is called and the recorded position variables are restored. The velocity and angular velocity of the object when entering the DropZone are returned to zero when the instrument is returned to its original position on the instrument tray to prevent the object continuing in its trajectory path when restored.

The primary function of the simulation is to provide a procedural training tool for pathology residents and clinicians to practice and review hospital autopsy procedure. Many of the methods covered in this study address the development of elements that form the foundation for a finished simulation. Specific steps within the renal block dissection procedure were selected to demonstrate the effectiveness of the simulation training and the potential for future iterations.

The visual aesthetic of the program was an important consideration in planning the user interaction. The interface was created with a minimalistic design that is intuitive to navigate and requires very little orientation to learn the controls. The environment consists of a white dome surrounding the user and a raised platform below the autopsy table. These elements decrease peripheral distraction and allow the user to become more completely immersed in the task at hand.

The program begins with the title screen and a series of user interface buttons. This screen serves as a starting point for each simulation and allows the user to select their desired experience from the user interface menus. The menu items remain present throughout the simulation to provide information on the user's progress and the option to change the simulation options. The main menu items are:

- Mode - The mode toggle displays the mode menu. The mode menu contains three toggle options:

- Instruction - Instruction mode takes the user through a step-by-step animated tutorial of the selected procedure using voice and text prompts. The instruction mode will be added in a later iteration of the program.

- Practice - Practice mode allows the user to explore the anatomy and practice the procedural steps with instructional prompts as needed.

- Testing - Testing mode removes instructional prompts and tests the user on their knowledge of the procedural steps. The testing mode is completed once the user has completed each of the procedural steps. The user will be tested on proper order of procedural steps and accuracy of cut placement to provide a more comprehensive view of the user's comprehension of the procedure. This feature will be available in future iterations of the application.

- Log - A log is created for each unique user to save any notes or findings the user records with each case. The log also keeps a record of past testing mode results and maps the user's competency over time. As pathologies are introduced in future iterations of the program, the log will help users recognize patterns and review cases. The log also contains the weights and measurements collected with each case. The user can choose to remove log entries or mark them for future review.

- Female/Male - The female and male toggles select the gender of the avatar displayed in the center of the UI menu. The avatar is a simplified model of the human body that provides orientation for the selected anatomy. The dissectable anatomy is the same for each avatar except for the pelvic block. When pressed, the female and male toggles display the block dissection panel containing six dissection blocks throughout the body:

- Heart - The heart has its own dissection procedure.

- Lungs - The lungs including associated airways.

- Hepatic Block - The hepatic block consists of the liver, esophagus, stomach, duodenum, gallbladder, pancreas, spleen, and all associated vessels, lymph nodes, and connective tissues.

- Renal Block - The renal block consists of the kidneys, ureters, bladder, aorta, and inferior vena cava.

- Pelvic Block - The pelvic block is the only dissection that differs between the female and male anatomy. The female pelvic block consists of the vagina, cervix, uterus, ovaries, and fallopian tubes. The male pelvic block consists of the prostate and seminal vesicles.

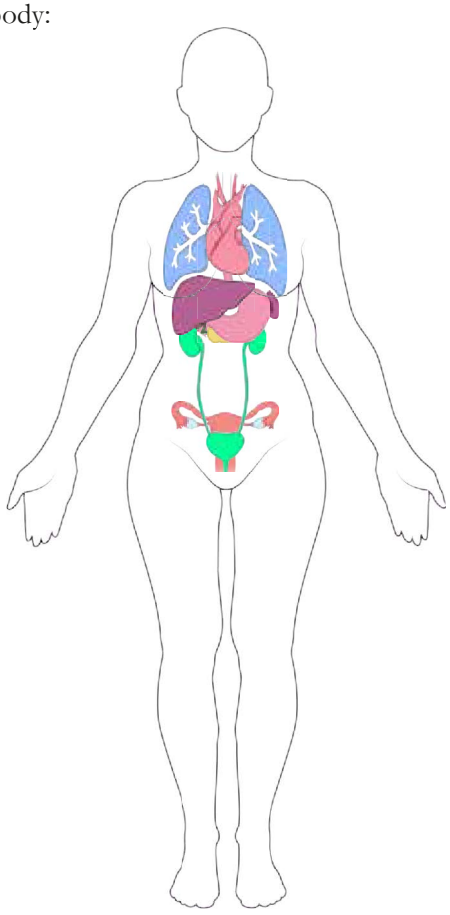

Figure 8. Female avatar displaying all anatomical blocks.

- Options - When pressed, the options toggle reveals the options panel containing the sound FX slider (active in instruction mode). The options panel also contains the controller configuration toggle, which displays the controller layout image in the center of the UI menu.

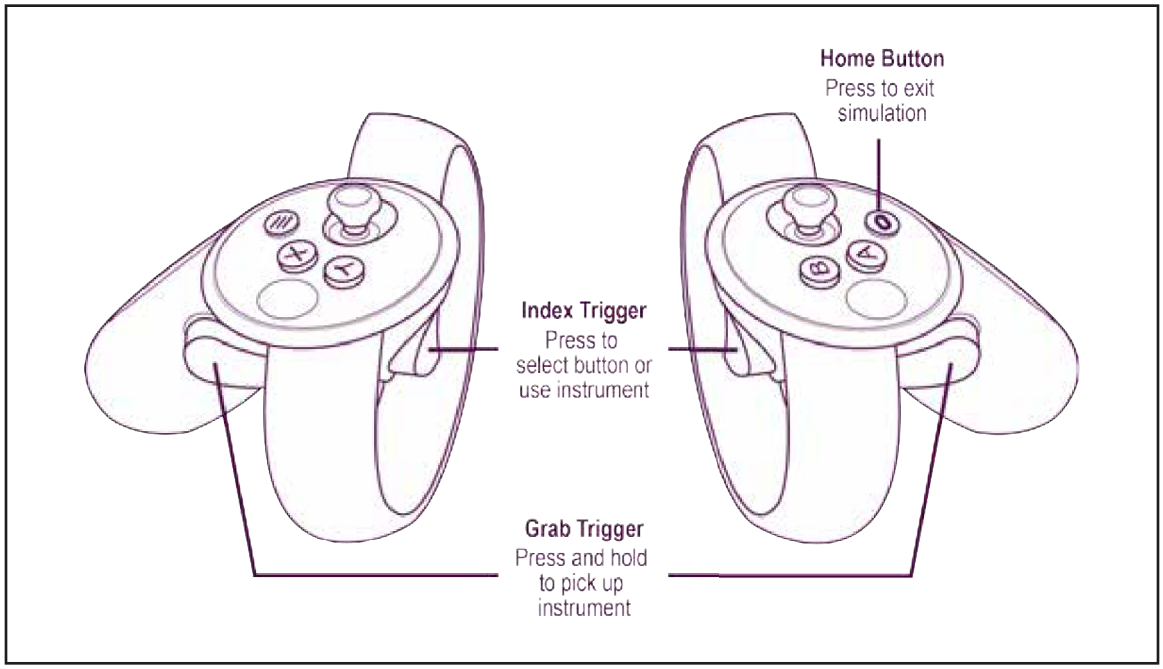

Figure 9. Controller button configuration.

Each of these panels allows the user to customize the simulation parameters to meet their needs. When a selection has been made from the mode menu, the female/male options, and the block dissection menu, the generate button becomes active which, when pressed, generates the simulation that meets the specified parameters. The user's virtual right hand has a light blue beam emitting from the tip of the index finger to assist the user in accurately interacting with the user interface toggles.

Currently the practice and renal block toggles are the only combination that will activate the generate button. When pressed, the generate button reveals the selected anatomy on the table in front of the user and a list of the available steps displayed at the center of the user interface menu. The generate button also converts to a reset button that will reset the current simulated anatomy to its initial position. As each procedural step is completed the associated menu item changes color and the

next step is highlighted. Those steps still under development are displayed to outline the complete procedure, but are grayed-out and cannot be selected.

The renal block dissection has been partially developed for this project and the instruction and testing modes are still in development. All unavailable options selected by the user will return a prompt indicating that the content is currently unavailable. A description of each menu item is included when the associated button is pressed to provide information about future iterations of the program.

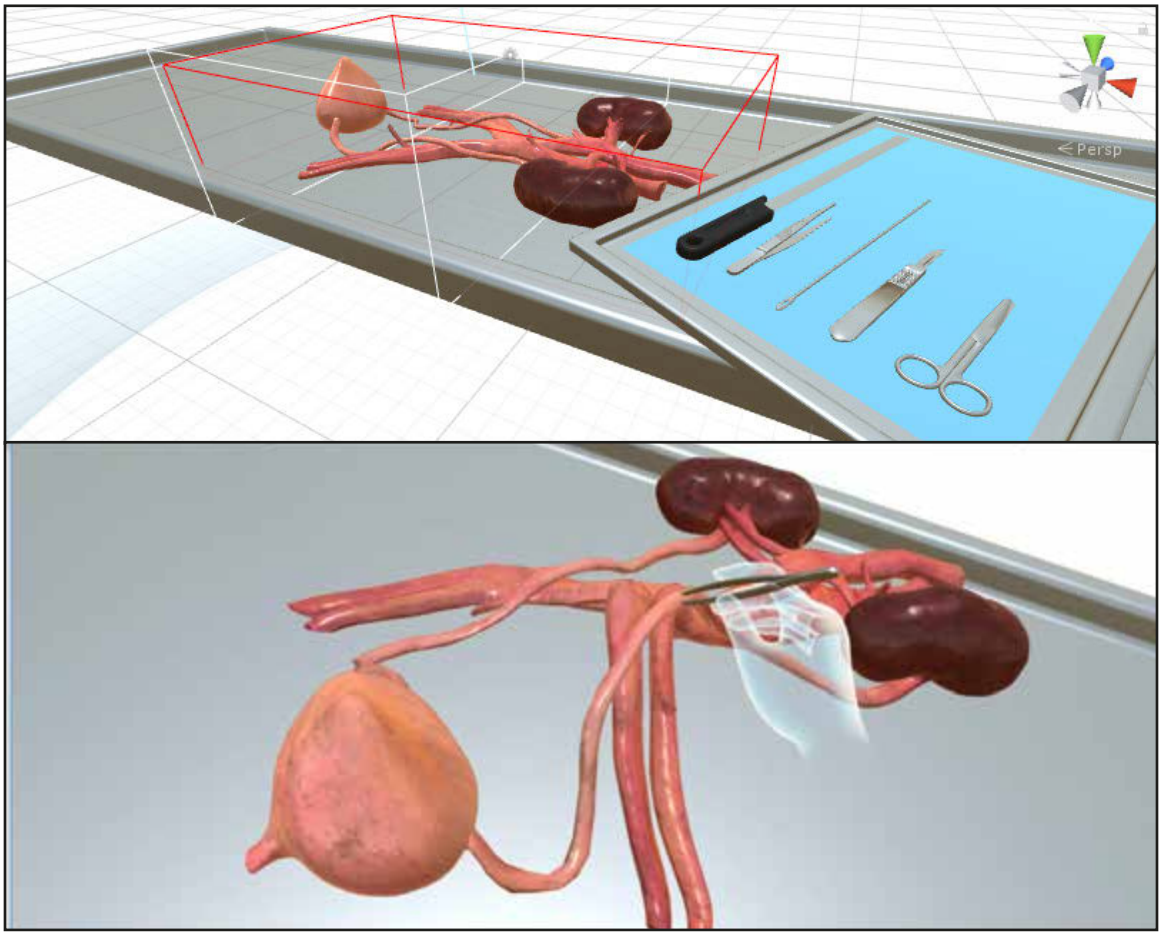

Figure 10. User interaction with virtual instruments and anatomy in Play mode.

Once the anatomy is displayed on the autopsy table, the user is able to interact with the anatomy by using the instruments displayed on the side table to the user's right. Each instrument becomes highlighted when contacted by the user's virtual hand. Highlighting indicates that the instrument

is available for use. The highlighted instrument is picked up by pressing the grab button on the controller located under the middle finger of the appropriate hand. Once picked up, the instrument is oriented in the proper position for use and the virtual hand models conform to reflect an accurate position on the instrument.

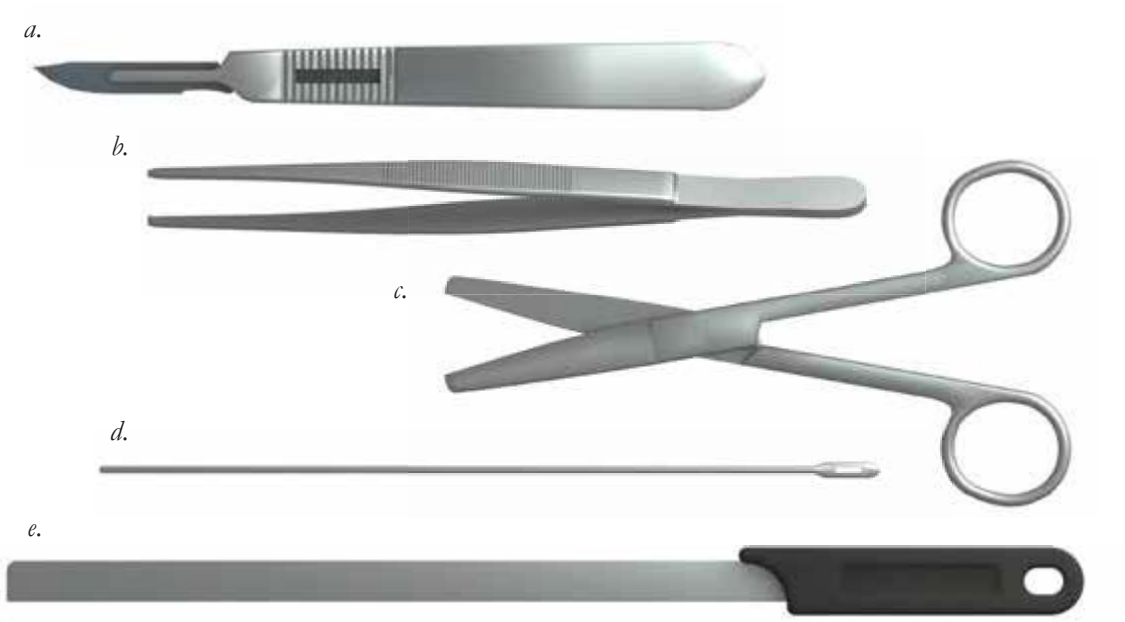

Figure 11. Instrument models (not to scale): scalpel (a), forceps (b), abdominal scissors (c), probe (d), autopsy knife (e).

Each instrument has a function to help the user perform the steps of the dissection procedures.

- Scalpel - The scalpel is modeled to represent a #74 handle and #60 blade commonly used in autopsy procedures. The scalpel is configured to make small, fine cuts.
- Forceps - The forceps' toothed tips grab the anatomical elements and are used to move or reposition the tissue.
- Abdominal scissors - The abdominal scissors are used to cut tissue without affecting the surrounding anatomy.
- Probe - The probe is introduced into tubular structures such as vessels or ureters to ensure patency. The probe is also used as a cutting guide for the autopsy knife.
- Autopsy knife - The autopsy knife is used to make large, planar cuts into the anatomy.

The images and models resulting from the work of this thesis will be partially found at **www.dhmedicalmedia.com**. Access to the models, scripts, and simulation can be granted by contacting the author at **danielwhermansen@gmail.com** or through the website of the Department of Art as Applied to Medicine at Johns Hopkins University School of Medicine: http://medicalart.johnshopkins.edu/.

Development of a finished version of the **Virtual Reality Guide to Hospital Autopsy** is being discussed in collaboration with the Pathology Department at Johns Hopkins University School of Medicine. The intended platform for initial release is the Oculus store with later expansions compatible with the HTC Vive and PSVR headsets.

Many capabilities and limitations of the current technology were noted throughout the development of the **Virtual Guide to Hospital Autopsy: Block Dissection of the Renal System**. Some of the technical challenges that were encountered are discussed below to assist other developers who undertake similar projects. Future iterations of the project are also discussed along with possible applications of this project in other healthcare applications.

General Considerations Regarding Virtual Reality in Medical Simulation

Some medical practitioners are skeptical of the benefits of simulation, stating that the intricacies of procedures may be too complex to for accurate simulation. Recent advances in VR technology have expanded the possibilities of its use in medical simulation, however some remaining limitations will need to be overcome before it gains widespread acceptance. The first obstacle to overcome is the association of VR with the gaming industry. Unfortunately, this is a stereotype that is reinforced by the fact that most content developed for consumer VR hardware is centered around gaming and entertainment. Furthermore, early adopters of VR technology often have a prior proclivity for gaming that encourages the marketing departments of VR companies to target this demographic.

Those medical education applications currently available for Oculus® and HTC Vive® fail to offer much more than a tablet based application. Applications like these, while initially engaging, quickly lose their appeal as a useful learning tool as they require the user to be disconnected from other study aids while using the headset. Most of the interactivity is limited to point and click functions to reveal information and in doing so fail to utilize the range of unique functionality offered by VR controllers. For this reason, it is recommended that medical simulations utilizing VR adopt some of the features commonly found in games.

Perhaps the most compelling feature of the VR experience is the sense of presence. If a user becomes so immersed in a simulation and reacts naturally to the virtual environment, the lessons learned will more readily translate into reproducible skills in real world environments. This is accomplished by creating realistic feedback to as many of the user's senses as possible. VR is primarily an audiovisual communication tool; however, it also utilizes the user's sense of spatial awareness. When combined with interactive controllers such as the Oculus Touch motion controllers, this sense

of presence within a virtual environment creates an almost ideal template for medical training.

A significant limiting factor in creating a realistic medical simulation using consumer ready hardware and game engines is the ability to render soft body tissue reaction. Several major medical simulation companies have created custom game engines that can handle the extensive computations required to render realistic soft tissues and recognize and respond accurately to user interaction, but most of these are housed in custom consoles that can be extremely cost prohibitive. Several options exist on the Unity® and Steam® asset stores that allow for soft body and cloth simulation, but each of these assets comes with its own limitations. However, utilizing a combination of assets allows one to simulate an approximation of accurate tissue reaction and set the stage to clearly depict the salient procedural steps.

Organ Block Considerations

The renal block underwent several iterations to find the best solution for the simulation. Initial tests utilized Unity's physics engine, but it was immediately evident that the soft body representations needed for vessels and ureters were not possible. A segmented version of the anatomy was created and linked with a series of hinge joints that performed reasonably well with Unity physics, but appeared disjointed while at rest and showed major break points when stretched. After several tests the model was rejected.

The next option utilized soft body and cloth assets from the Unity asset store. Each of the following assets was tested and eventually rejected based on their inability to simulate interactive tissue-like reactions.

- Caronte FX® - Caronte FX is a powerful renderer for Unity animations. It can render the soft body reactions necessary for the simulation, but is not intended to do so in real-time interactions. It is also capable of creating baked animation prefabs, which can be introduced into the simulation on trigger events, but this removes a level of control from the user.

- Obi Advanced Cloth Simulation® - Obi creates cloth surfaces which can be open or closed objects with a variety of properties. A model of the renal system was created with the vessels

and ureters as Obi Cloth elements, however the anatomy needed to be closed and a level of inflation added to each element or the vessels and ureters would deflate to a flat, flexible surface. The inflation caused the models to increase in size beyond their predefined limits.

The greatest benefit of the Obi Cloth solution was the ability to attach points of the anatomy to non-Obi objects such as the kidneys and bladder. This allowed the cloth surfaces to follow the collider bodies they were attached to and any attached anatomy would follow if allowed by the stretch limits. The connections did not always function perfectly, however, and would often degrade into a mass of intersecting geometry that would crash the system.

Any object with an Obi Cloth component attached also became "ungrabbable" in Play mode. This presented problems while attempting to perform the procedural steps of cutting and probing the ureters. Eventually, the ureter cloth surfaces were abandoned and replaced with a chain of capsules linked by a series of hinge joints. Each ureter was attached to a kidney and the bladder. This allowed the ureters to be grabbed, but the divisions along the ureter chain were clearly evident during manipulation. When attempting to cut the ureters, a glitch would cause the hinge joints of the chain to explode and send the capsules and connected anatomy flying in all directions, often resulting in a system crash.

The kidneys were pre-divided into 3 sections; a top, bottom and middle. Each section had a rigidbody component and kidney material applied. The top and bottom sections contained fixed and hinge joints to the middle section, allowing each kidney to move as a complete unit until cut with the autopsy knife thus destroying the fixed joint. This allowed the user to separate the kidney sections to expose the inner anatomy. The problem with this approach was that the separate pieces of kidneys failed to recognize each other and would flip around the hinge joints when freed. Limits were added to the hinge joints with limited success, but the pieces still failed to recognize the cloth surface anatomy and resulted in an entanglement of organs.

- Turbo Slicer 2® - The Turbo Slicer 2 asset worked well for dividing the rigidbody objects, but did not function on the uFlex soft body objects used in the final model.

Future Iterations

The Virtual Reality Guide to Hospital Autopsy will cover the entire block dissection procedure consisting of the following: Heart, lungs, hepatic, renal, and pelvic blocks. The final simulation will also include instruction and testing modes. The mode hierarchy will walk the user through a step-by-step tutorial of the procedure, allow the user to practice the techniques on their own, and test their knowledge of the steps and the accuracy of their technique. Based on the user's performance, they will receive a score and a rank represented by 1 to 5 stars. Each testing result will be stored in the individual's user log, which will track performance over time. This will allow the user to challenge themselves to improve on past scores and recognize areas that need additional focus.

An additional feature included in future versions of the simulation will be pathologies within the organic blocks. This will serve to instruct the user on the features of various pathologies and add an additional level of complexity to the testing mode. As each abnormality is discovered and observed, the user will be able to save an image and description of the pathology in their log, allowing the user to later review pathologies they have discovered. Any pathologies missed by the user in the testing mode will be noted and negatively impact the final score.

The ultimate goal of any healthcare practice is to improve patient care and outcome. The practice of performing hospital autopsies continues to provide valuable insight into the nature of pathologies that resulted in the death of a patient. Improvements throughout the healthcare system can be implemented based on the findings of a post mortem examination, many of which can prevent future loss of life. Medical simulations, such as the one in this study, are designed to provide the training necessary to safely perform complex procedures like a hospital autopsy. The fundamental methodology developed for this project can be applied to a wide range of applications to healthcare. The adoption of interactive simulations into healthcare training will directly impact patient outcomes and ultimately save lives.

By creating an immersive virtual reality simulation of a hospital autopsy, pathology residents and clinicians can review and practice the procedural techniques without risk to a patient. While the simulation developed in this study serves only as a conceptual model, the success experienced warrants further development into a finished training application covering the full range of techniques employed throughout the hospital autopsy procedure.

APPENDIX A: OVERLORD USER INTERFACE SCRIPT

```
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class Overlord : MonoBehaviour {
    private HashSet<string> actives = new HashSet<string>();
    //All submenus to hide/show on button selection
    private GameObject mode_submenu;
      GameObject inactivate(string name) {
       var go = GameObject.Find(name);
       go.SetActive(false);
       return go;
    }
    void Start() {
       mode_submenu = inactivate("ModeTG");
    }
    public void ToggleChange(Toggle t) {
       if (t.isOn)
          this.actives.Add(t.name);
       else
          this.actives.Remove(t.name);
       if (actives.Contains("Renal_Btn") && actives.Contains("Practice_Btn")) {
          generate_btn.SetActive(true);
       }
       else {
          generate_btn.SetActive(false);
       }
    }
}
```

Appendix B: ButtonStyler UI script

```
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.EventSystems;
public class ButtonStyler : MonoBehaviour, IPointerEnterHandler, IPointerExitHandler {
    private Text txt;
    private Image bkg;
    public Color32 purple = new Color32(0x54, 0x02, 0x4f, 0xff);
    private Sprite bkg_on;
    private Sprite bkg_off;
    public void StartStyler() {
        txt = this.gameObject.GetComponentInChildren<Text>();
        bkg = this.gameObject.GetComponentInChildren<Image>();
        bkg.color = this.purple;
        bkg_on = Resources.Load<Sprite>("__blank");
        bkg_off = Resources.Load<Sprite>("SquareFrame");
        bkg.overrideSprite = bkg_off;
        PointerExit(false);
    }
    public void OnPointerEnter(PointerEventData pd) {
        txt.color = Color.white;
        bkg.overrideSprite = bkg_on;
    }
    public virtual void OnPointerExit(PointerEventData pd) {
        PointerExit(false);
    }
    public void PointerExit(bool stayOn) {
        if (stayOn) {
            txt.color = Color.white;
            bkg.overrideSprite = bkg_on;
        }
        else {
            txt.color = this.purple;
            bkg.overrideSprite = bkg_off;
        }
    }
}
```

Appendix C: Custom Hand Pose script

```
using System.Collections.Generic;
using UnityEngine;
using VRTK;
public class LocalAvatarHandController : MonoBehaviour {
    Dictionary<string, Transform> hands = new Dictionary<string, Transform>();
    Dictionary<string, string> target_to_hand = new Dictionary<string, string>();
    OvrAvatar ovr;
    VRTK_InteractGrab left_grab, right_grab;
    VRTK_InteractUse left_use, right_use;
    private GameObject pickups;
    private GameObject scissors;
    void Start () {
        target_to_hand["Long Knife"] = "_Knife";
        target_to_hand["Scalpel"] = "_Scalpel";
        target_to_hand["Pickups"] = "_PickupsOpen";
        target_to_hand["Scissors"] = "_ScissorsOpen";
        target_to_hand["Probe"] = "_Probe";
        foreach (var hand in Resources.LoadAll<Transform>("Hands"))
            hands[hand.name] = hand;
        ovr = this.GetComponentInChildren<OvrAvatar>();
left_grab = GameObject.Find("LeftController").GetComponent<VRTK_InteractGrab>();
right_grab = GameObject.Find("RightController").GetComponent<VRTK_InteractGrab>();
left_use = left_grab.GetComponent<VRTK_InteractUse>();
right_use = right_grab.GetComponent<VRTK_InteractUse>();
left_grab.ControllerGrabInteractableObject += new ObjectInteractEventHandler(onGrab);
left_grab.ControllerUngrabInteractableObject += new ObjectInteractEventHandler(onUngrab);
right_grab.ControllerGrabInteractableObject += new ObjectInteractEventHandler(onGrab);
right_grab.ControllerUngrabInteractableObject += new ObjectInteractEventHandler(onUngr
ab);
right_use.ControllerUseInteractableObject += new ObjectInteractEventHandler(onUse);
right_use.ControllerUnuseInteractableObject += new ObjectInteractEventHandler(onUnuse);
left_use.ControllerUseInteractableObject += new ObjectInteractEventHandler(onUse);
left_use.ControllerUnuseInteractableObject += new ObjectInteractEventHandler(onUnuse);
        pickups = GameObject.Find("Pickups");
        scissors = GameObject.Find("Scissors");
    }
```

```
private void onGrab(object sender, ObjectInteractEventArgs e) {
    if (!target_to_hand.ContainsKey(e.target.name))
        return;
    if (e.controllerIndex == 0)
        ovr.LeftHandCustomPose = hands["HandLeft" + target_to_hand[e.target.name]];
    else
        ovr.RightHandCustomPose = hands["HandRight" + target_to_hand[e.target.name]];
    if (e.target == scissors)
        activateAllBut(scissors, "Closed");
    else if (e.target == pickups)
        pickups.GetComponent<PickupsTrigger>().onGrab();
}
private void onUngrab(object sender, ObjectInteractEventArgs e) {
    if (e.controllerIndex == 0)
        ovr.LeftHandCustomPose = null;
    else if (e.controllerIndex == 1)
        ovr.RightHandCustomPose = null;
    if (e.target == pickups)
        pickups.GetComponent<PickupsTrigger>().onUngrab();
}
private void activateAllBut(GameObject g, string name) {
    for (int i = 0; i < g.transform.childCount; ++i) {
        var ch = g.transform.GetChild(i);
        ch.gameObject.SetActive(ch.name != name);
    }
}
private void onUse(object sender, ObjectInteractEventArgs e) {
    if (e.target == pickups) {
        activateAllBut(pickups, "Open");
        pickups.GetComponent<PickupsTrigger>().onUse();
    }
    if (e.target == scissors) {
        if (e.controllerIndex == 0)
            ovr.LeftHandCustomPose = hands["HandLeft_ScissorsClosed"];
        else
            ovr.RightHandCustomPose = hands["HandRight_ScissorsClosed"];
        activateAllBut(scissors, "Open");
```

```
        }
    }
    private void onUnuse(object sender, ObjectInteractEventArgs e) {
        if (e.target == pickups) {
            activateAllBut(pickups, "Closed");
            pickups.GetComponent<PickupsTrigger>().onUnuse();
        }
        if (e.target == scissors) {
            if (e.controllerIndex == 0)
                ovr.LeftHandCustomPose = hands["HandLeft_ScissorsOpen"];
            else
                ovr.RightHandCustomPose = hands["HandRight_ScissorsOpen"];
            activateAllBut(scissors, "Closed");
        }
    }
}
```

APPENDIX D: FORCEPS PICKUP SCRIPT

```
using System.Collections.Generic;
using System.Linq;
using UnityEngine;
using uFlex;
struct IdMass {
    public int idx;
    public float inv_mass;
    public IdMass(int idx, float inv_mass) : this() {
        this.idx = idx;
        this.inv_mass = inv_mass;
    }
}
public class PickupsTrigger : FlexProcessor {
    private List<IdMass> locked = new List<IdMass>();
    private Collider ball;
    private bool try_pick_up = false, try_drop = false;
    private void Start() {
        ball = GetComponents<Collider>().ToList().Where(c => c.enabled).First();
```

```
    }
    public void onGrab() {
    }
    public void onUngrab() {
    }
    public void onUse() {
        try_pick_up = true;
    }
    public void onUnuse() {
        try_drop = true;
    }
    public override void PostContainerUpdate(FlexSolver solver, FlexContainer cntr,
FlexParameters parameters) {
        if (try_pick_up) {
            try_pick_up = false;
            locked = cntr.m_particles.ToList().Select((v, i) => new IdMass(i, v.invMass))
                .Where(v => Vector3.Distance(cntr.m_particles[v.idx].pos, ball.bounds.center) < 1.0f)
                .ToList();
        }
        else if (try_drop) {
            try_drop = false;
            locked.Select(l => cntr.m_particles[l.idx].invMass = l.inv_mass);
            locked.Clear();
        }
        else if (locked.Count() > 0) {
            var locked_ctr = locked.Select(l => cntr.m_particles[l.idx].pos)
                            .Aggregate((lhs, rhs) => lhs + rhs) / locked.Count();
            var delta = ball.bounds.center - locked_ctr;
            foreach (var l in locked) {
                cntr.m_particles[l.idx].pos += delta;
                cntr.m_velocities[l.idx] = delta / Time.fixedTime;
            }
        }
    }
}
```

APPENDIX E: INSTRUMENT DROPZONE SCRIPT

```
using System.Collections.Generic;
using System.Linq;
using UnityEngine;
struct ToolItems {
    public GameObject obj;
    public Vector3 pos;
    public Quaternion rot;
    public ToolItems(GameObject gameObject, Vector3 position, Quaternion rotation) : this() {
        this.obj = gameObject;
        this.pos = position;
        this.rot = rotation;
    }
}
public class OnDroppedTool : MonoBehaviour {
    private List<ToolItems> tools;
      void Start () {
        var inst = GameObject.Find("Instruments");
        tools = inst.GetComponentsInChildren<Transform>().ToList<Transform>()
            .Where(tx => tx.parent == inst.transform)
            .Select(t => new ToolItems(t.gameObject, t.position, t.rotation)).ToList<ToolItems>();
      }
    private void OnTriggerEnter(Collider other) {
        if (!other.transform.parent)
            return;
        var dropped_tools = tools.Where(tool => tool.obj == other.transform.parent.
gameObject);
        if (dropped_tools.Count() > 0) {
            var dropped = dropped_tools.First();
            dropped.obj.transform.position = dropped.pos;
            dropped.obj.transform.rotation = dropped.rot;
            var rb = dropped.obj.GetComponent<Rigidbody>();
            rb.velocity = Vector3.zero;
            rb.angularVelocity = Vector3.zero;
        }
    }
}
```

1. Hill RB, Anderson R. "The recent history of the autopsy." *Archives of Pathology and Laboratory Medicine* 1996; 120:702-712. PMID: 8718895

2. Shojania KG, Burton EC. "The vanishing nonforensic autopsy." *The New England Journal of Medicine* 2008; 358:873-875. PMID: 18305264

3. Davies DJ, Graves DJ, Landgren AJ, Lawrence CH, Lipsett J, MacGregor DP, Sage MD. "The decline of the hospital autopsy: a safety and quality issue for healthcare in Australia." *The Royal College of Pathologists of Australasia* 2004; 180 (6): 281-285

4. The American Board of Pathology. Combined Anatomic Pathology and Clinical Pathology (AP/CP). *Requirements for Certification* 2015; http://www.abpath.org/index.php/to-become-certified/requirements-for-certification?layout=edit&id=156

5. Lateef F. "Simulation-based learning: Just like the real thing." *Journal of Emergencies, Trauma, and Shock* 2009; 3(4): 348–352

6. Hutchins GM, Clingman C. "An Introduction to Autopsy Technique." *College of American Pathologists* 1991

7. Oculus Developer Center. Unity Packages. *Downloads* 2016; https://developer.oculus.com/downloads/unity/

8. Oculus Developer Center. Platform SDK Developer Guide. *Getting Started Guide,* 2016; https://developer3.oculus.com/documentation/platform/latest/concepts/book-pgsg/

Daniel Hermansen was born in Orem, Utah on April 19, 1986. Shortly thereafter, his family moved to Cedar City, Utah where he graduated from Cedar High School in 2004. Following graduation, Daniel chose to serve a full-time two year religious mission for The Church of Jesus Christ of Latter Day Saints and was called to serve in the Rome, Italy mission. Upon his return, Daniel matriculated at Southern Utah University where he graduated with honors in 2012 with a Bachelor of Science in Biology/Zoology and a minor in Illustration. Daniel worked as a freelance illustrator and graphic designer throughout his education and following graduation. After applying to several Physician Assistant graduate programs without success, he decided to continue his education by earning certifications in phlebotomy and as an advanced emergency medical technician. It was during this coursework that he learned about the field of medical illustration. While continuing to work at the Primary Children's Hospital and the University of Utah Hospital, Daniel completed the portfolio requirements for entrance into a medical illustration graduate program.

In August of 2015, he entered the Medical and Biological Illustration program in the Department of Art as Applied to Medicine at Johns Hopkins University School of Medicine in Baltimore, Maryland. Daniel is currently a candidate for a Master of Arts in Medical and Biological Illustration. Following graduation, he is looking to pursue a career developing virtual and augmented reality applications for medical simulation.